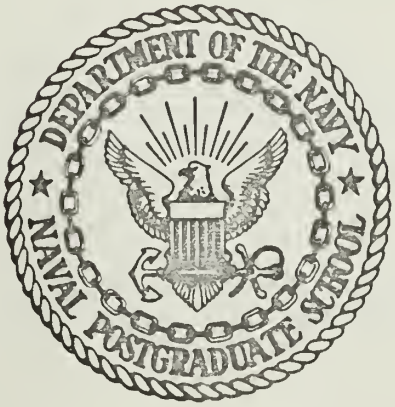
A COMPARISON OF METHODS FOR GENERATING MULTIVARIATE NORMAL RANDOM VECTORS

by

Norman Lee Slezak

United States
Naval Postgraduate School



THESIS

A COMPARISON OF METHODS FOR GENERATING MULTIVARIATE NORMAL RANDOM VECTORS

by

Norman Lee Slezak

September 1970

A Comparison of Methods for Generating Multivariate Normal Random Vectors

by

Norman Lee Slezak
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1960

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1970

ABSTRACT

Three methods for generating outcomes on multivariate normal random vectors are presented. A comparison is made to determine which method requires the least computer execution time and memory space when utilizing the IBM 360/67. All methods use as a basis a standard Gaussian random number generator. Results of the comparison study indicate that the method based on triangular factorization of the covariance matrix generally requires less memory space and computer time than the other two methods.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

Frequently it is desired to generate a sample of vectors derived from a specified multivariate normal distribution. For example, this need arises in simulation studies in which the model contains stochastic variates that are distributed according to a multivariate normal. Since a computer is frequently used to produce the sample vectors, the memory space and computer time requirements become important cost factors when considering various methods for "generating" random vectors.

The problem of "generating" random vectors was discussed in 1948 by Herman Wold [Ref. 1]. Wold describes a method for construction of samples from a multivariate normal distribution. Wold used a method based on triangularization of the covariance matrix. In 1962, Ernest M. Scheuer and David S. Stoller wrote a paper on the generation of multivariate normal random vectors [Ref. 2]. They considered a method based on matrix equations and a method based on conditional distributions. The first method uses a technique similar to that of Wold's. Their conclusion was that the mathematics involved in the first method was simpler than that of the second method so the generation of random vectors is best accomplished by the matrix equation technique.

The objective of this paper is to examine the above two methods, hereafter called "triangularization" and

"conditioning" respectively, and to discuss a third method which is referred to as the "rotation" method. Computer programs for each method are presented and a comparison of memory space requirements and execution times on the IBM 360/67 computer is made for the three methods. The format for making this comparison study is similar to that used by M. E. Muller in his paper on univariate normal generators [Ref. 3].

The following notation is used in this study: X denotes a random variable whereas x is a real variable. A vector is denoted by a bar under the letter. Listed below are two examples of the above notation:

$$\underline{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} \qquad \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

In the discussion of the three methods, generators for $N(\underline{0},\Sigma)$ distributions are considered, where Σ is a general (positive definite symmetric) matrix. There is no loss of generality in assuming the mean vector to be the zero vector since a random vector $\underline{Y}$ distributed $N(\underline{u},\Sigma)$ may be generated by first generating $\underline{X}$ distributed $N(\underline{0},\Sigma)$ and then taking $\underline{Y} = \underline{X} + \underline{u}$.

II. METHODS

A. ROTATION METHOD

Given a positive definite symmetric p x p matrix Σ, the function

$$f(\underline{x};\Sigma) = (2\pi)^{-\frac{1}{2}p} \, |\Sigma|^{-\frac{1}{2}} \, e^{-\frac{1}{2}\underline{x}'\Sigma^{-1}\underline{x}} \qquad (1)$$

is a p-variate normal density. A random vector $\underline{X}$ having this distribution has mean $\underline{0}$ and variance covariance matrix Σ [Ref. 4]. Denote the distribution law corresponding to the density function (1) by $N(\underline{0},\Sigma)$. If $\underline{X}$ (with p components) is distributed $N(\underline{0},\Sigma)$, then $\underline{Y} = C\underline{X}$ is distributed $N(\underline{0},C\Sigma C')$, if C is nonsingular [Ref. 4]. The development of the rotation method is based on this fact. If a p x p matrix C is found such that $C\Sigma C'$ yields a diagonal matrix, then elements on the diagonal of $C\Sigma C'$ are the variances of the corresponding components of $\underline{Y}$. Since the covariance values (off diagonal elements of $C\Sigma C'$) are zero, the components of $\underline{Y}$ are independent.

For a symmetric matrix Σ, there exists an orthogonal matrix C such that $C\Sigma C' = D$ where D contains the eigen values of Σ on its diagonal [Ref. 5]. Associated with the eigen values are eigen vectors which make up the rows of the matrix C. Since $\underline{Y} = C\underline{X}$ is $N(\underline{0},C\Sigma C')$, and there exists a C such that $C\Sigma C'$ is the diagonal matrix D, it follows that the components of $\underline{Y}$ are independent normal variates with zero mean and variance values as given by the

corresponding eigen values of Σ. These variates may be generated using a univariate normal random number generator. A method of generating the random vector $\underline{Y}$ is thus easily established. To obtain a generator for the random vector $\underline{X}$, a transformation is made using $\underline{X} = C^{-1}\underline{Y}$. Since C is an orthogonal matrix, this simplifies to the equation $\underline{X} = C'\underline{Y}$. Standard matrix multiplication of the orthogonal matrix C' and sample vectors generated on the random vector $\underline{Y}$ yields the desired random samples of the vector $\underline{X}$.

B. CONDITIONING METHOD

A multivariate probability density function f may be written as:

$$f(x_1,\ldots,x_n) = f(x_1|x_2,\ldots,x_n)\ f(x_2|x_3,\ldots,x_n) \ldots f(x_{n-1}|x_n)\ f(x_n) \qquad (2)$$

where $f(x_j|x_i\ldots)$ denotes a conditional density function. If X_n can be generated from $f(x_n)$, then X_{n-1} may be generated by conditioning on X_n. X_{n-2} may then be generated by conditioning on X_n and X_{n-1}. In a similar manner, all the components of $\underline{X}$ may be generated. In what follows, this general approach is applied to multivariate normal distributions.

Given that the vector $\underline{X}$ is partitioned into two subvectors $\underline{X}^{(1)}$ and $\underline{X}^{(2)}$ and that the covariance matrix of $\underline{X}$ is correspondingly partioned into Σ_{11}, Σ_{12}, and Σ_{22}, then it can be shown [Ref 4] that the conditional distribution

of $\underline{X}^{(1)}$ given $\underline{X}^{(2)} = \underline{x}^{(2)}$ is normal with mean $\Sigma_{12}\,\Sigma_{22}^{-1}\,\underline{x}^{(2)}$ and covariance matrix $\Sigma_{11} - \Sigma_{12}\,\Sigma_{22}^{-1}\,\Sigma_{21}$. Several definitions are introduced at this time as they are required in what follows.

Let

$$\Sigma^{(k)} = \begin{bmatrix} \sigma_{k,k} & \sigma_{k,k+1} & \cdots & \sigma_{k,n} \\ \sigma_{k+1,k} & \sigma_{k+1,k+1} & \cdots & \sigma_{k+1,n} \\ \vdots & \vdots & & \vdots \\ \sigma_{n,k} & \sigma_{n,k+1} & \cdots & \sigma_{n,n} \end{bmatrix}$$

and let $\Sigma_{ij}^{(k)}$ denote the cofactor of σ_{ij} in $\Sigma^{(k)}$; similarly Σ_{ij} denotes the cofactor of σ_{ij} in Σ. Consider now the mean and variance of the distribution determined by $f(x_k | x_{k+1},\ldots,x_n)$. Suppose the covariance matrix Σ is partioned as follows:

$$\Sigma = \left[\begin{array}{c|c} \Sigma_{11} = [\sigma_{k,k}] & \Sigma_{12} = [\sigma_{k,k+1} \quad \sigma_{k,k+2} \quad \cdots \quad \sigma_{k,n}] \\ \hline \Sigma_{21} = \begin{bmatrix} \sigma_{k+1,k} \\ \sigma_{k+2,k} \\ \vdots \\ \sigma_{n,k} \end{bmatrix} & \Sigma_{22} = \begin{bmatrix} \sigma_{k+1,k+1} & \sigma_{k+1,k+2} & \cdots & \sigma_{k+1,n} \\ \sigma_{k+2,k+1} & \sigma_{k+2,k+2} & \cdots & \sigma_{k+2,n} \\ & & & \\ \sigma_{n,k+1} & \sigma_{n,k+2} & \cdots & \sigma_{n,n} \end{bmatrix} \end{array}\right] \qquad (3)$$

The variance of $f(x_k|x_{k+1},\ldots,x_n)$ is given by:

$$V(X_k|x_{k+1},\ldots,x_n) = \Sigma_{11} - \Sigma_{12}\,\Sigma_{22}^{-1}\,\Sigma_{21} \qquad (4)$$

To simplify this expression, the following theorem is used [Ref. 4]: $|\Sigma| = |\Sigma_{11} - \Sigma_{12}\,\Sigma_{22}^{-1}\,\Sigma_{21}|\cdot|\Sigma_{22}|$. The square matrix Σ_{22} is nonsingular since the covariance matrix Σ is positive definite. The variance (4) can be written as:

$$V(X_k|x_{k+1},\ldots,x_n) = \frac{|\Sigma^{(k)}|}{|\Sigma^{(k+1)}|} \qquad (5)$$

The mean of $f(x_k|x_{k+1},\ldots,x_n)$ is given by:

$$E(X_k|x_{k+1},\ldots,x_n) = \Sigma_{12}\,\Sigma_{22}^{-1}\,\underline{x}^{(2)}$$

Direct substitution of components of the partitioned covariance matrix (3) into the latter expression yields:

$$E(X_k|x_{k+1},\ldots,x_n) = [\sigma_{k,k+1}\ \sigma_{k,k+2}\ \cdots\ \sigma_{k,n}](\Sigma^{(k+1)})^{-1}\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix}$$

A simplification of this expression can be made as follows. First, examine $(\Sigma^{(k+1)})^{-1}$. By using standard matrix algebra, it can be shown that:

$$(\Sigma^{(k+1)})^{-1} =$$

$$\frac{1}{|\Sigma^{(k+1)}|}\begin{bmatrix} \Sigma^{(k+1)}_{k+1,k+1} & -\Sigma^{(k+1)}_{k+2,k+1} & \cdots & (-1)^{n+k+1}\Sigma^{(k+1)}_{n,k+1} \\ -\Sigma^{(k+1)}_{k+1,k+2} & \Sigma^{(k+1)}_{k+2,k+2} & \cdots & (-1)^{n+k+2}\Sigma^{(k+1)}_{n,k+2} \\ \vdots & \vdots & & \vdots \\ (-1)^{k+1+n}\Sigma^{(k+1)}_{k+1,n} & (-1)^{k+2+n}\Sigma^{(k+1)}_{k+2,n} & \cdots & (-1)^{2n}\Sigma^{(k+1)}_{n,n} \end{bmatrix}$$

Next, perform the matrix multiplication

$$[\sigma_{k,k+1} \quad \sigma_{k,k+2} \quad \cdots \quad \sigma_{k,n}]\,[\Sigma^{(k+1)}]^{-1}.$$

This yields a vector of dimension 1 x (n - k + 1). The transpose of this vector is the product of the factor $\frac{1}{|\Sigma^{(k+1)}|}$ and

$$\begin{bmatrix} [+\Sigma^{(k+1)}_{k+1,k+1}\sigma_{k,k+1} - \Sigma^{(k+1)}_{k+1,k+2}\sigma_{k,k+2} \cdots (-1)^{k+1+n}\Sigma^{(k+1)}_{k+1,n}\sigma_{k,n}] = a_1 \\ [-\Sigma^{(k+1)}_{k+2,k+1}\sigma_{k,k+1} + \Sigma^{(k+1)}_{k+2,k+2}\sigma_{k,k+2} \cdots (-1)^{k+2+n}\Sigma^{(k+1)}_{k+2,n}\sigma_{k,n}] = a_2 \\ \vdots \\ [(-1)^{k+1+n}\Sigma^{(k+1)}_{n,k+1}\sigma_{k,k+1} \quad (-1)^{k+2+n}\Sigma^{(k+1)}_{n,k+2} \cdots (-1)^{2n}\Sigma^{(k+1)}_{n,n}\sigma_{k,n}] = a_3 \end{bmatrix}$$

(6)

Multiply the vector (6) by $[x_{k+1},\ldots,x_n]$ and the following results are obtained:

$$E(X_k|x_{k+1},\ldots,x_n) = \frac{1}{|\Sigma^{(k+1)}|}\,[(a_1)x_{k+1} + (a_2)x_{k+2} \ldots + (a_3)x_n]$$

where (a_1) indicates the first element of the vector (6), (a_2) indicates the second element, etc. Upon close inspection, it is apparent that (a_1) is equivalent to $-\Sigma^{(k)}_{k+1,k}$. In the same manner, (a_2) is $-\Sigma^{(k)}_{k+2,k}$, ..., and (a_3) is $-\Sigma^{(k)}_{n,k}$. The mean of $f(x_k|x_{k+1},\ldots,x_n)$ can thus be represented as:

$$E(X_k|x_{k+1},\ldots,x_n) = -\frac{1}{|\Sigma^{(k+1)}|}\,[\Sigma^{(k)}_{k+1,k}\,x_{k+1} + \Sigma^{(k)}_{k+2,k}\,x_{k+2}\ldots+\Sigma^{(k)}_{n,k}x_n] \tag{7}$$

The use of these expressions in generating the components of $\underline{X}$ is now described. Consider the transformation $X_k = \sigma_k\,Y_k + \mu_k$, $k = 1,2,\ldots\ n$, where the Y's are independent normal variates with zero mean and unit variance. Using expressions (5) and (7) for σ_k and μ_k respectively, this transformation may be given explicitly as:

$$X_n = Y_n\,\sqrt{\sigma_{n,n}} \tag{8}$$

$$X_k = \left[\frac{|\Sigma^{(k)}|}{|\Sigma^{(k+1)}|}\right]^{\frac{1}{2}} Y_k - \frac{1}{|\Sigma^{(k+1)}|}\left[\Sigma^{(k)}_{k+1,k}X_{k+1} + \Sigma^{(k)}_{k+2,k}X_{k+2}\ldots+\Sigma^{(k)}_{n,k}X_n\right] \tag{9}$$

for $k = 1,\ldots,n-1$. Use of the equations (8) and (9) therefore provides a method of generating samples of a random vector $\underline{X}$, using univariate generators for the Y_k's.

C. TRIANGULARIZATION METHOD

The covariance matrix, designated as Σ, is a symmetric positive definite matrix. It can be written as $\Sigma = TT'$ where T is a lower triangular matrix [Ref. 6],

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix} = \begin{bmatrix} t_{11} & 0 & \cdots & 0 \\ t_{21} & t_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{bmatrix} \begin{bmatrix} t_{11} & t_{21} & \cdots & t_{n1} \\ 0 & t_{22} & \cdots & t_{n2} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & t_{nn} \end{bmatrix}$$

$\underline{X}$ is obtained by making the non-singular transformation $\underline{X} = T\underline{Y}$ where $\underline{Y}$ is distributed $N(\underline{0},I)$. By previous remarks, $\underline{X}$ is distributed $N(\underline{0},TT')$. The matrix T can be obtained from Σ by using the so called "square root method" [Ref. 6]. Equations for t_{ij} in terms of σ_{ij} are as follows:

$$t_{i1} = \sigma_{i1} / \sqrt{\sigma_{11}} \qquad ; \quad 1 \leq i \leq n$$

$$t_{ii} = \sqrt{\sigma_{ii} - \sum_{p=1}^{i-1} t_{ip}^2} \qquad ; \quad 1 < i \leq n$$

$$t_{ij} = \frac{1}{\sigma_{jj}} \left[\sigma_{ij} - \sum_{p=1}^{j-1} t_{ip}\, t_{jp}\right] \qquad ; \quad 1 < j < i \leq n$$

$$t_{ij} = 0 \qquad ; \quad i < j \leq n$$

The vector $\underline{Y}$ consists of independent normal components with zero mean and unit variance. These may be generated using a univariate normal random number generator. Standard matrix multiplication of T and samples of $\underline{Y}$ yields samples of the random vector $\underline{X}$.

III. METHOD EVALUATION

Computer runs were obtained using each of the three methods, with various size covariance matrices. Data concerning these runs is compiled in Tables I, II, and III. A brief description of the procedure employed to obtain execution time and memory space requirements for the tables follows. The computer programs can be subdivided into three basic portions. The first portion consists of the data that must be entered into the computer. The second portion consists of all required steps, prior to the use of the basic univariate random number generator, necessary to implement one of the three methods described in this study. The third portion consists of the standard Gaussian random number generator and those steps necessary for generating samples of the random vector, $\underline{X}$, such that a matrix is filled with one thousand random vectors. Two times were considered when evaluating the programs: the "set-up" time and the "repetition" time. These times correspond to execution time for the second and the third portion respectively of the computer program. Similarly, the memory space requirements shown in the tables make use of two numbers. The first number provides an indication of the amount of space required for the computer programs for each of the three methods. This number includes the subroutine EIGEN in the rotation method and the function GRN

in all cases. The second number is the total space required for the program plus any external function used such as square roots, absolute values, exponentials, input and output devices, etc. The second number varies as to the computer system being used and in this case the computer was the IBM 360/67. The headings on two columns of Tables I, II, and III are labeled "Specific Conditioning" and "Specific Triangularization" respectively. These refer to computer programs written specifically for 2 x 2 and 3 x 3 covariance matrices, and are adaptations of the general programs.

Based on the data contained in Tables I, II, and III, the best method to use for generating random vectors from covariance matrices dimensioned greater than 3 x 3 appears to be the triangularization method. This method requires less memory space and considerably less execution time than the other two methods. If the user is interested in generating samples from a bivariate or trivariate normal distribution, then either a specific conditioning or triangularization method can be used. Both methods require about the same amount of execution time and their space requirements are essentially the same.

All methods use, as a basis, the univariate normal random number generator. Therefore, in order to establish a lower bound on times required in generating normal random vectors, the time required to generate normal random numbers was obtained. These numbers should serve as lower

bounds on the times required to generate a sample of one thousand repetitions of a 10, 5, 3, and 2 element vector respectively. The lower bounds obtained in this way are:

10,000 numbers - 1.06740 seconds
5,000 numbers - 0.53209 seconds
3,000 numbers - 0.32302 seconds
2,000 numbers - 0.20332 seconds

A specialized covariance matrix, (identity matrix), was used as an input to all three methods to provide the least cumbersome, mathematically speaking, of all matrices that would be encountered. The results of this test are tabulated in Table III. Each method maintained its overall ranking with respect to memory space and time requirements. The rotation method displayed a sharp decrease in set-up time which is reasonable as eigen values are easier (faster) to compute in this case. In general, each of the three methods required essentially the same amount of time for 1000 repetitions using a typical covariance matrix as for the specialized covariance matrix.

TABLE I

MEMORY SPACE REQUIREMENTS

The first number is the amount of core space required for the computer program and the second number is the total amount of core space required in execution of the program. This second number includes, for example, external functions and input-output devices. Each space requirement is in bytes.

Matrix Size	Conditioning	Triangular-ization	Rotation
2 x 2	11,792/37,168	9,528/33,864	11,952/36,288
3 x 3	15,880/41,256	13,568/37,904	16,008/40,344
5 x 5	24,160/49,536	21,712/46,048	24,192/48,528
10 x 10	45,424/70,800	42,336/66,672	44,992/69,328
- - - -	Specific Cond.	Specific Triang.	- - - -
2 x 2	8,752/33,088	8,744/33,080	- - - -
3 x 3	12,976/37,312	12,960/37,296	- - - -

TABLE II

EXECUTION TIME (TYPICAL COVARIANCE MATRIX)

The first number is the set-up time, and the second number is the repetition time. Each execution time is in seconds.

Matrix Size	Conditioning	Triangular-ization	Rotation
2 x 2	.00273/.33985	.00106/.33800	.00496/.37666
3 x 3	.00741/.55383	.00145/.52411	.01398/.62023
5 x 5	.03975/1.06335	.00247/.99554	.06117/1.23357
10 x 10	.70306/2.75560	.00795/2.52280	.41652/3.72289
- - - -	Specific Cond.	Specific Triang.	- - - -
2 x 2	.00091/.25246	.00101/.27487	- - - -
3 x 3	.00101/.39530	.00109/.39572	- - - -

TABLE III

EXECUTION TIME (SPECIALIZED COVARIANCE MATRIX I)

The first number is the set-up time, and the second number is the repetition time. Each execution time is in seconds.

Matrix Size	Conditioning	Triangular-ization	Rotation
2 x 2	.00278/.34299	.00122/.33509	.00171/.36376
3 x 3	.00793/.55869	.00161/.54223	.00234/.60250
5 x 5	.03214/1.06025	.00260/.98449	.00403/1.19545
10 x 10	.33795/2.69337	.00733/2.42645	.01162/3.41616
- - - -	Specific Cond.	Specific Triang.	- - - -
2 x 2	.00104/.25215	.00078/.24921	- - - -
3 x 3	.00135/.37903	.00096/.38795	- - - -

APPENDIX A

DISCUSSION OF THE COMPUTER PROGRAMS

All three computer programs were written in Fortran IV. They are not optimal, i.e., improvements to the programs are possible. The results of these improvements could change the value of memory space requirements and program execution times, but should not change the comparative ranking among the three methods.

Two univariate normal random number generators that were available for use in the present study were GAUSS and GRN. The subroutine GAUSS is part of the IBM scientific subroutine package and is based on the mean value theorem. The function GRN was compiled by the Naval Postgraduate School staff, and is based on the G. Marsaglia technique [Ref. 7]. The execution time of GRN is approximately ten times faster than GAUSS, and for this reason it was used in all the programs. The rotation method uses the subroutine EIGEN which is also available as part of the IBM scientific subroutine package. Three subroutines, (DTERM, REDMAT, and COFACT), were developed for use in the conditioning method. The subroutine DTERM computes the determinant of an n-square matrix. The subroutine COFACT removes the elements from the i^{th} row and j^{th} column of an n-square matrix (Z), and provides the (n - 1)-square matrix that is required in the computation of the cofactor Z_{ij}.

The subroutine REDMAT reduces a matrix to the desired size. It is capable of providing the user with a $\Sigma^{(k)}$ from an n x n dimension matrix where $k \leq n$.

Each method was evaluated using four different size matrices as listed in Tables I, II, and III. These were selected as being typical of the range of matrix sizes that might be encountered in practice. However, the programs were written in general terms so there are no program imposed restrictions on matrix size.

As mentioned previously, specific computer programs were written for the 2 x 2 and the 3 x 3 covariance matrices using the conditioning and triangularization method. This was done in an attempt to evaluate the possible reduction in memory space and execution time under those obtained using the general programs with input dimensions of 2 and 3. Of the three methods considered, these two appeared to be adaptable to simpler, more concise programs for these small dimensions. Writing specific programs for covariance matrices dimensioned greater than 3 x 3 would become difficult, and a general program would probably have to be used. No attempt was made to write a specific program for the rotation method since computation of eigen values and vectors are involved, and there were no apparent means of reducing the computation time under that of using the general subroutine EIGEN.

The computer programs, used in generating multivariate normal random vectors for each of the three methods, are included as a portion of this study.

THIS PROGRAM UTILIZES THE ROTATION METHOD TO GENERATE MULTIVARIATE NORMAL RANDOM VECTORS.

```
      DIMENSION Z(3,3), R(3,3), XA(3,1000), B(6), Y(3), T(3)
      READ (5,1) N
1     FORMAT (I5)
      DO 2 I = 1,N
2     READ (5,3) (Z(I,J), J = 1,N)
3     FORMAT (3F8.0)
      MV = 0
      K = 1
      DO 5 J = 1,N
      DO 4 I = 1,J
      B(K) = Z(I,J)
4     K = K + 1
5     CONTINUE
      LB = K - 1
      CALL EIGEN (B,R,N,MV)
      MA = 1
      JA = 1
      DO 6 LA = 1,N
      Y(JA) = B(MA)
      Y(JA) = SQRT(Y(JA))
      JA = JA + 1
6     MA = MA + JA
      IDUMMY = 0
      DO 10 MT = 1,1000
      DO 7 JP = 1,N
      X = (GRN(IDUMMY))*Y(JP)
7     T(JP) = X
      DO 9 I = 1,N
      SUM = C.0
      DO 8 J = 1,N
8     SUM = SUM + R(I,J) * T(J)
9     XA(I,MT) = SUM
10    CONTINUE
      STOP
      END
```

```
      THIS PROGRAM UTILIZES THE TRIANGULARIZATION METHOD
      TO GENERATE MULTIVARIATE NORMAL RANDOM VECTORS.



      DIMENSION Z(3,3), C(3,3), XV(1000,3), T(3)
      READ (5,1) N
1     FORMAT (I8)
      READ (5,2) ((Z(I,J), J = 1,N), I = 1, N)
2     FORMAT (3F8.0)
      DO 4 I = 1,N
      C(I,1) = Z(I,1)/SQRT (Z(1,1))
      IF ( I + 1 .GT. N) GO TO 4
      JE = I + 1
      DO 3 JB = JE,N
      C(I,JB) = 0.0
3     CONTINUE
4     CONTINUE
      DO 10 I = 2,N
      IF (I - 3) 8,5,5
5     JT = I - 1
      DO 7 J = 2,JT
      SUN = 0.0
      JD = J - 1
      DO 6 KA = 1,JD
      SUN = SUN + C(I,KA) * C(J,KA)
6     CONTINUE
      C(I,J) = (Z(I,J) - SUN)/C(J,J)
7     CONTINUE
8     SUM = 0.0
      JF = I - 1
      DO 9 K = 1,JF
      SUM = SUM + C(I,K) * C(I,K)
9     CONTINUE
      C(I,I) = SQRT (Z(I,I) - SUM)
10    CONTINUE
      IDUMMY = 0
      DO 14 JP = 1,1000
      DO 11 JC = 1,N
      X = GRN(IDUMMY)
11    T(JC) = X
      DO 13 IT = 1,N
      SUMM = 0.0
      DO 12 J = 1,IT
12    SUMM = SUMM + C(IT,J) * T(J)
      XV(JP,IT) = SUMM
13    CONTINUE
14    CONTINUE
      STOP
      END
```

```
      THIS PROGRAM UTILIZES THE CONDITIONAL METHOD TO
      GENERATE MULTIVARIATE NORMAL RANDOM VECTORS.



      DIMENSION Z(3,3), A(3,3), B(3,3), Y(3), R(3), T(3),
     1XA(1000,3), ZJ(3,3)
      READ (5,1) N
1     FORMAT (I4)
      READ (5,2) ((Z(I,J), J = 1,N), I = 1,N)
2     FORMAT (3F8.0)
      M = N - 1
      DO 6 L = 1,M
      K = N - L
      LT = N - K + 1
      CALL REDMAT (Z,N,LT,K,A)
      IR = K + 1
      DO 3 IV = IR, N
      NV = LT - 1
      IF = IV - K + 1
      IG = 1
      CALL COFACT (LT,A,B,NV,IF,IG)
      CALL DTERM (NV, B, D, NV)
      IT = IF + IG
3     ZJ(K,IV) = ((-1)**IT) * D
      CALL DTERM (LT,A,D,LT)
      DNUM = D
      DO 4 IS = 1,N
      DO 4 JS = 1,N
      A(IS,JS) = 0.0
      B(IS,JS) = 0.0
4     CONTINUE
      CALL REDMAT (Z,N,NV,IR,A)
      CALL DTERM(NV,A,D,NV)
      T(K) = D
      DO 5 IC = 1,N
      DO 5 JC = 1,N
      A(IC,JC) = 0.0
5     CONTINUE
6     R(K) = SQRT(DNUM/T(K))
      AD = SQRT(Z(N,N))
      IDUMMY = 0
      DO 10 NOP = 1,1000
      DO 7 NO = 1,N
      X = GRN(IDUMMY)
7     Y(NO) = X
      XA(NOP,N) = AD * Y(N)
      LA = N - 1
      DO 9 NORM = 1,LA
      SUMN = 0.0
      K = N - NORM
      LEE = K + 1
      DO 8 NSYL = LEE,N
8     SUMN = ZJ(K,NSYL) * XA(NOP,NSYL) + SUMN
9     XA(NOP,K) = R(K) * Y(K) - (SUMN/T(K))
10    CONTINUE
      STOP
      END
```

```
      SUBROUTINE DTERM (N,A,D,M)
      DIMENSION A(M,M)
      DD = 1.0
      DO 16 L = 1,N
      KP = 0
      Z = 0.0
      DO 12 K = L,N
      IF (Z - ABS(A(K,L))) 11,12,12
11    Z = ABS(A(K,L))
      KP = K
12    CONTINUE
      IF (L - KP) 13,15,15
13    DO 14 J = L,N
      Z = A(L,J)
      A(L,J) = A(KP,J)
14    A(KP,J) = Z
      DD = - DD
15    IF (L - N) 16,20,20
16    LP1 = L + 1
      DO 19 K = LP1 , N
      IF (A(K,L)) 17,19,17
17    RATIO = A(K,L)/A(L,L)
      DO 18 J = LP1,N
18    A(K,J) = A(K,J) - RATIO * A(L,J)
19    CONTINUE
20    DO 21 K = 1,N
21    DD = DD * A(K,K)
      D = DD
      RETURN
      END


      SUBROUTINE REDMAT (Z,N,LZ,IB,A)
      DIMENSION Z(N,N), A(LZ,LZ)
      I = 0
      DO 23 L = 1,N
      IF (L .LT. IB) GO TO 23
      I = I + 1
      J = 0
      DO 22 K = 1,N
      IF (K .LT. IB) GO TO 22
      J = J + 1
      A(I,J) = Z(L,K)
22    CONTINUE
23    CONTINUE
      RETURN
      END


      SUBROUTINE COFACT (NA,Z,B,NV,IK,IL)
      DIMENSION Z(NA,NA), B(NV,NV)
      I = 0
      DO 25 K = 1,NA
      IF (K .EQ. IK) GO TO 25
      I = I + 1
      J = 0
      DO 24 L = 1,NA
      IF (L .EQ. IL) GO TO 24
      J = J + 1
      B(I,J) = Z(K,L)
24    CONTINUE
25    CONTINUE
      RETURN
      END
```

BIBLIOGRAPHY

1. Wold, H., Tracts for Computers, No. XXV, Cambridge Press, 1955.

2. Scheuer, E. M. and Stoller, D. S., "On the Generation of Normal Random Vectors," Technometrics, v. 4, no. 2, p. 278-281, May 1962.

3. Muller, M. E., "A Comparison of Methods for Generating Normal Deviates on Digital Computers," Journal of the Association for Computing Machinery, v. 6, no. 3, p. 376-383, July 1959.

4. Anderson, T. W., An Introduction to Multivariate Statistical Analysis, p. 17-19, App. A, Wiley, 1960.

5. Graybill, F. A., An Introduction to Linear Statistical Models, p. 5, v. I, McGraw-Hill, 1961.

6. Graybill, F. A., Introduction to Matrices with Applications in Statistics, p. 298, Wadsworth, 1969.

7. Marsaglia, G., "A Fast Procedure for Generating Normal Random Variables," Communications of the ACM, v. 7, no. 1, p. 4-10, January 1964.

8. Howe, J. E., The Generation of Random Numbers from Various Probability Distributions, Masters Thesis, Naval Postgraduate School, Monterey, 1965.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department of Operations Analysis, Code 55 Naval Postgraduate School Monterey, California 93940	1
4.	Associate Professor D. R. Barr, Code 55 bn Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
5.	LCDR N. L. Slezak, USN Box 34 Milligan, Nebraska 68406	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3 REPORT TITLE

A COMPARISON OF METHODS FOR GENERATING MULTIVARIATE NORMAL RANDOM VECTORS

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; September 1970

5. AUTHOR(S) (First name, middle initial, last name)

Norman Lee Slezak, Lieutenant Commander, United States Navy

6. REPORT DATE
September 1970

7a. TOTAL NO. OF PAGES
30

7b. NO. OF REFS
8

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

Three methods for generating outcomes on multivariate normal random vectors are presented. A comparison is made to determine which method requires the least computer execution time and memory space when utilizing the IBM 360/67. All methods use as a basis a standard Gaussian random number generator. Results of the comparison study indicate that the method based on triangular factorization of the covariance matrix generally requires less memory space and computer time than the other two methods.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
MULTIVARIATE NORMAL DISTRIBUTION						
RANDOM NUMBER GENERATOR						
NORMAL DISTRIBUTION						
MULTIVARIATE NORMAL GENERATOR